

**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK  
MODUL IV  
PEWARISAN TUNGGA (INHERITANCE)**



Oleh:

Ramadhan Wijaya

2211102208

IF-10-M

**S1 TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## I. DASAR TEORI

inheritance adalah sebuah konsep dari generalization atau juga bisa disebut warisan. jadi mudahnya inheritance adalah prosedur untuk mewariskan data dan method dari class ke class. karena mewariskan data dan method dari class ke class maka class yang mewariskan disebut sebagai superclass sedangkan class yang diwarisi disebut subclass. untuk mempermudah, kita analogikan dengan hewan dan kucing. Hewan adalah superclass sedangkan kucing adalah subclass karena hewan adalah generalisasi dari kucing.

untuk mendeklarasikan inheritance pada java dilakukan seperti dibawah ini :

### a. Kelas induk

```
public class BangunDatar {  
  
    float luas(){  
        System.out.println("Menghitung laus bangun datar");  
        return 0;  
    }  
  
    float keliling(){  
        System.out.println("Menghitung keliling bangun datar");  
        return 0;  
    }  
}
```

Keterangan : seperti pendeklarasian class pada umumnya

### b. Kelas anak

```
package inheritance;  
  
public class Persegi extends BangunDatar {  
    float sisi;  
}
```

```
public class Lingkaran extends BangunDatar{  
  
    // jari-jari lingkaran  
    float r;  
  
}
```

```
public class PersegiPanjang extends BangunDatar {  
    float panjang;  
    float lebar;  
}
```

Keterangan : pendeklarasian class anak adalah dengan tambahan *extends* *namasubclass*

Fungsi adanya sistem inheritance ini adalah dapat membuat kelas baru yang mewarisi atribut dan metode dari kelas yang sudah ada. Ini memungkinkan Anda untuk menggunakan kembali kode yang telah ditulis sebelumnya tanpa perlu menuliskannya ulang. Seperti konsep fungsi dan prosedur tapi bedanya pada saat memanggil superclass dengan berada di baris awal pendeklarasian, sedangkan jika fungsi yang dipanggil di fungsi lain pendeklarasiannya di dalam badan fungsi atau procedure.

Konstruktor dan destructor dalam sistem inheritance Dalam pemrograman Java, konstruktor adalah metode khusus yang dipanggil saat objek suatu kelas dibuat. Konstruktor digunakan untuk menginisialisasi objek dan menetapkan nilai awal ke dalam variabel-variabelnya. Sebaliknya, Java tidak memiliki "destructor" seperti yang ada dalam bahasa pemrograman lain seperti C++ atau Python. Java menggunakan sistem garbage collection untuk mengelola memori dan menangani penghapusan objek yang tidak lagi digunakan, sehingga tidak memerlukan destructor.

Kelas turunan **DAPAT** mengakses setiap public member kelas dasar, kelas lain juga **DAPAT** mengakses member kelas dasar secara langsung. Kelas turunan **TIDAK DAPAT** mengakses private member kelas dasar, kelas lain juga **TIDAK DAPAT** mengakses member kelas dasar secara langsung. Kelas turunan **DAPAT** mengakses setiap protected member kelas dasar, tetapi kelas lain **TIDAK DAPAT** mengakses member kelas dasar secara langsung.

Overriding atau sering pula disebut dengan redefinisi adalah kemampuan suatu kelas anak untuk memodifikasi (mendefinisikan kembali) data dan method dari kelas induknya. Proses ini akan mengubah data method dari keduanya, kelas induk dan kelas anaknya. Alasan mengapa dilakukan overriding antara lain jika akan dilakukan perubahan secara menyeluruh, baik jumlah maupun tipe parameter maupun behaviour pemrosesan datanya. Overriding dapat juga dilakukan jika akan dilakukan perubahan hanya untuk menambahkan behaviour khusus yang dimiliki hanya oleh kelas anak tersebut. Yang perlu diperhatikan dalam melakukan overriding adalah modifier penentu aksesibilitas data dan methodnya yakni private, public atau protected.

1. Public Mengijinkan kelas dan sub kelas dari package manapun untuk mengaksesnya.
2. Private Membatasi akses hanya untuk kelas itu sendiri dan objek yang diinstans darinya.
3. Protected Akses hanya diberikan kepada kelas itu sendiri dan sub kelas yang diturunkan darinya.

Jadi , jelas bahwa overriding hanya bisa dilakukan untuk data dan method yang memiliki modifier penentu aksesibilitas public dan protected.

## II. GUIDED

DemoPoint.cpp

```
//Ramadhan Wijaya
//22111102208
#include <iostream>
#include <conio.h>

using namespace std;

// Kelas Point sebagai kelas induk
class Point
{
public:
    Point(float = 0.0, float = 0.0); // konstruktor default
    void cetakPoint();              // akan di-redefinisi
    di kelas anak
    ~Point();                        // destruktur
protected:
    float x, y;
};

Point::Point(float a, float b)
{
    cout << "Konstruktor Point dijalankan " << endl;
    x = a;
    y = b;
}

void Point::cetakPoint()
{
    cout << "Point : " << '[' << x << ", " << y << ']' <<
endl;
}

Point::~~Point()
{
    cout << "Destruktor Point dijalankan" << endl;
}

// Kelas Circle : Merupakan kelas turunan dari kelas Point
class Circle : public Point
{
public:
    Circle(float r = 0.0, float x = 0.0, float y = 0.0); //
konstruktor
```

```

        void cetakPoint();                                //
redefinisi fungsi kelas induk
        ~Circle();                                        //
destruktor
private:
        float radius;
};

// memanggil konstruktor kelas induk dan inisialisasi data
member warisan
Circle::Circle(float r, float a, float b) : Point(a, b)
{
        cout << "Konstruktor Circle dijalankan" << endl;
        radius = r;
}
void Circle::cetakPoint()
{
        Point::cetakPoint();
        cout << "Circle : radiusnya " << radius << endl;
}

// destruktor
Circle::~~Circle()
{
        cout << "Destruktor Circle dijalankan " << endl;
}

int main()
{
        Point p(1.1, 2.2); // Pembentukan Objek Point
        cout << endl;
        Circle circle1(6.5, 8.2, 1.9); // Pembentukan Objek
Circle
        circle1.cetakPoint(); // Pemanggilan Method
        cout << endl;
        Circle circle2(10, 5, 5);
        circle2.cetakPoint();
        getch();
        return 0;
}

```

Output :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Exe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'
Konstruktor Point dijalankan

Konstruktor Point dijalankan
Konstruktor Circle dijalankan
Point : [8.2, 1.9]
Circle : radiusnya 6.5

Konstruktor Point dijalankan
Konstruktor Circle dijalankan
Point : [5, 5]
Circle : radiusnya 10
Destruktor Circle dijalankan
Destruktor Point dijalankan
Destruktor Circle dijalankan
Destruktor Point dijalankan
Destruktor Point dijalankan
PS D:\SEMESTER_3\Praktikum_PBO\MODUL4\C++>
```

Keterangan :

program diatas berisikan dua class yaitu class Point sebagai kelas induk dan class circle sebagai kelas anak. pada program diatas juga tertera pada setiap class mau class induk atau anak tetap sama sama membuat destructor karena destructor dari class induk tidak bisa diakses oleh kelas anak. pada program utamanya terlihat perbedaan cara pendeklarasian class induk yaitu Point dan class anak yaitu Circle, pada class anak bisa terlihat bisa memasukan nilai yang sama dan nilai tambahan dari nilai yang ada di classnya sendiri, pernyataan ini terlihat pada program dibawah ini

```
// memanggil konstruktor kelas induk dan inisialisasi data
member warisan
Circle::Circle(float r, float a, float b) : Point(a, b)
{
    cout << "Konstruktor Circle dijalankan" << endl;
    radius = r;
}
```

Terlihat Class Circle memanggil konstruktor kelas induk Point(a,b)

Dengan cara pemanggilan menggunakan :: dan :

Diakhir output terlihat jelas ada program destructor yang berjalan karena pada program c++ tidak terdapat sistem garbage collection.

### **DemoPoint.java**

Class induknya

```
/**
 *
 * @author ramadhan wijaya
 * @2211102208
 * @IF-10M
 */

/*Kelas Induk */
public class Point {

    protected float x, y;
    // Konstruktor kelas Induk

    public Point(float a, float b) {
        System.out.println("Konstruktor Point dijalankan ");
        x = a;
        y = b;
    }
    // Method kelas Induk

    public void cetakPoint() // akan di-redefinisi di kelas anak
    {
        System.out.println("Point : [" + x + ", " + y + "]");
    }

    /**Main Method*/
    public static void main(String args[]) {
        // Membuat instans dari kelas anak dan memanggil method yang dimilikinya
        Circle circle1 = new Circle(6.5f, 8.2f, 1.9f);
        circle1.cetakPoint();
        System.out.println(" ");
        Circle circle2 = new Circle(10, 5, 5);
        circle2.cetakPoint();
    }
}
```

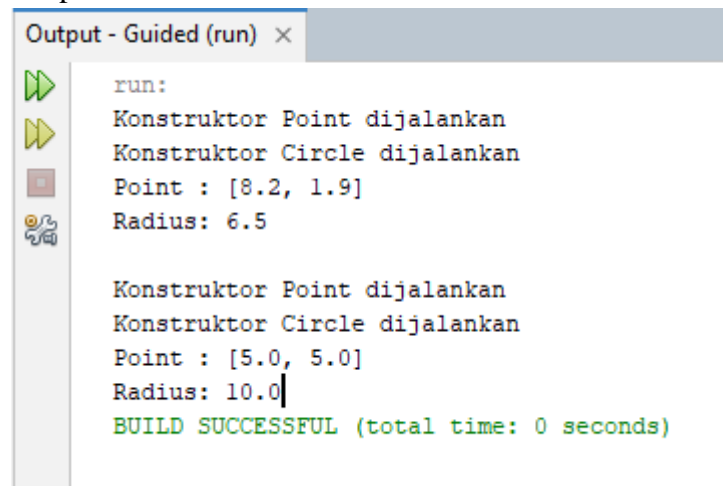


## Kelas anak

```
/**
 * @author ramadhan wijaya
 * @2211102208
 * @IF-10M
 */
/* Kelas Anak */
public class Circle extends Point {
    private float radius;
    // Konstruktor kelas Anak memanggil konstruktor kelas Induk
    public Circle(float r, float a, float b) {
        super(a, b);
        radius = r;
        System.out.println("Konstruktor Circle dijalankan ");
    }
    // Method yang memanggil method Induk dari kelas induknya dengan
    menggunakan keyword super
    public void cetakPoint() // redefinisi fungsi kelas induk
    {
        super.cetakPoint();
        System.out.println("Radius: " + radius);
    }

    /**Main Method*/
    public static void main(String args[]) {
        // Membuat instans dari kelas anak dan memanggil method yang
        dimilikinya
        Circle circle1 = new Circle(6.5f, 8.2f, 1.9f);
        circle1.cetakPoint();
        System.out.println(" ");
        Circle circle2 = new Circle(10, 5, 5);
        circle2.cetakPoint();
    }
}
```

Output :



```
Output - Guided (run) x
run:
Konstruktor Point dijalankan
Konstruktor Circle dijalankan
Point : [8.2, 1.9]
Radius: 6.5

Konstruktor Point dijalankan
Konstruktor Circle dijalankan
Point : [5.0, 5.0]
Radius: 10.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan : sama seperti penjelasan program c++ sebelumnya hanya saja pada java penulisan class child, penanda class child, dan tidak adanya destructor lah yang membedakan, oleh karena itu tidak ada ouput destructor berjalan di akhir ouput.

Demo2.java

```
package ProjectSpeaks;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
class Dog extends Mammal {

    public void speak() {
        System.out.println("Arf! Arf!");
    }

}
```

```
package ProjectSpeaks;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
```

```
class Duck extends Mammal {  
  
    public void speak() {  
        System.out.println("Quack! Quack!");  
    }  
}
```

```
package ProjectSpeaks;  
  
/**  
 *  
 * @author ramadhan wijaya  
 * 2211102208  
 */  
class Horse extends Mammal {  
  
    public void speak() {  
        System.out.println("Whinny! Whinny!");  
    }  
}
```

```
package ProjectSpeaks;  
  
/**  
 *  
 * @author ramadhan wijaya  
 * 2211102208  
 */  
class Mammal {  
  
    public String name;  
  
    public void sleep() {  
  
        System.out.println("ZZZZ ZZZZZZ ZZZZ");  
  
    }  
}
```

```
package ProjectSpeaks;  
  
/**  
 *
```

```

* @author ramadhan wijaya
* 2211102208
*/
class Owl extends Mammal {

    public void speak() {
        System.out.println("Whoo! Whoo!");
    }

    public void sleep() {
        System.out.println(" "); // Owl makes no noise when sleeping
    }
}

```

```

package ProjectSpeaks;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
class MorleySafer extends Mammal {

    public void speak() {

        System.out.println("Can I ask you a few questions about " + "your
1987 tax statement?");
    }
}

```

```

package ProjectSpeaks;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
class MikeWallace extends Mammal {

    public void speak() {
        System.out.println("Can you honestly look the American " +
"people in the eye and say that?");
    }
}

```

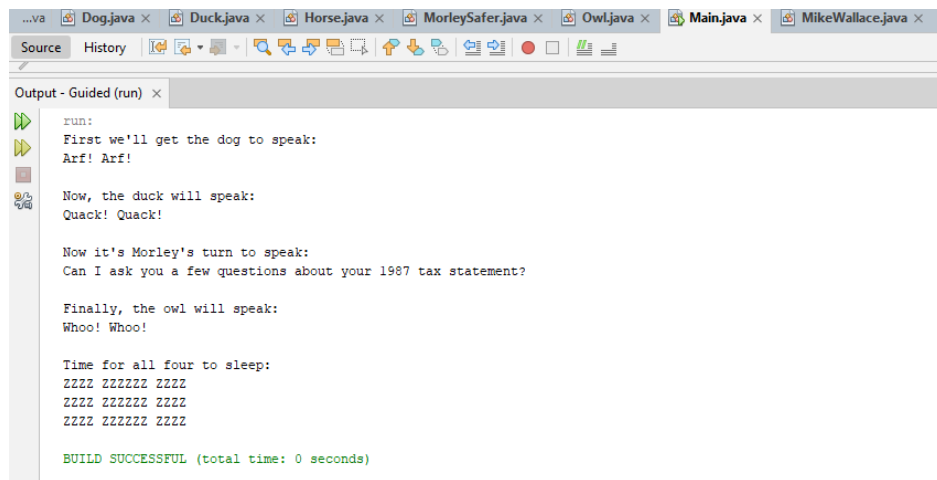
```
}
```

```
package ProjectSpeaks;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class Main {

    public static void main(String[] arguments) {
        Dog guffy = new Dog();
        Duck donald = new Duck();
        MorleySafer morley = new MorleySafer();
        Owl woodsy = new Owl();
        guffy.name = "Guffy";
        donald.name = "Donald";
        morley.name = "Morley Safer";
        woodsy.name = "Woodsy";
        System.out.println("First we'll get the dog to speak:");
        guffy.speak();
        System.out.println(" ");
        System.out.println("Now, the duck will speak:");
        donald.speak();
        System.out.println(" ");
        System.out.println("Now it's Morley's turn to speak:");
        morley.speak();
        System.out.println(" ");
        System.out.println("Finally, the owl will speak:");
        woodsy.speak();
        System.out.println(" ");
        System.out.println("Time for all four to sleep:");
        guffy.sleep();
        donald.sleep();
        morley.sleep();
        woodsy.sleep();
    }
}
```

Output :



```
run:
First we'll get the dog to speak:
Arf! Arf!

Now, the duck will speak:
Quack! Quack!

Now it's Morley's turn to speak:
Can I ask you a few questions about your 1987 tax statement?

Finally, the owl will speak:
Whoo! Whoo!

Time for all four to sleep:
ZZZZ ZZZZZZ ZZZZ
ZZZZ ZZZZZZ ZZZZ
ZZZZ ZZZZZZ ZZZZ

BUILD SUCCESSFUL (total time: 0 seconds)
```

### Keterangan :

Program diatas berisi banyak class child, satu class parent yaitu `mamal.java` dan satu class main untuk memanggil semua yg dibutuhkan. Pada class parent `mamal.java` terdapat variable `name` dan method `sleep()`; sedangkan pada kebanyakan kelas child yaitu `dog.java`, `duck.java`, `horse.java`, dan lain lain terdapat method `speaks()`; Dimana setiap hewan yang bersuara memiliki suara yang berbeda beda. Pada class child juga memiliki sintaks `extends` `mamal` yang berfungsi untuk menjelaskan bahwa program itu memiliki class parent yaitu class `mamal`. Pada program utama berisikan cara pembuatan class baru, penginisialisasian variable `name` yang Dimana variable `name` terdapat di class parent

Perhatikan main dari kelas **Speak.java** diatas!!

Jelaskan mengapa objek `guffy`, `donald`, `morley`, dan `woodsdy` dapat mengakses atribut `name` dan method `sleep` dari kelas `Mammal` !! **Bahas dalam laporan!**

Jawab : karena terdapat sintaks `extends` `mamal` pada setiap class class tersebut yang menandakan bahwa class class tersebut adalah class child yang dimana mendapatkan variable `name` dan method `sleep()`; dari class parentnya yaitu class `mamal`. Contohnya pada class `dog` dibawah ini

```
class Dog extends Mammal {
    public void speak() {
        System.out.println("Arf! Arf!");
    }
}
```

### III. UNGUIDED

#### IV. UNGUIDED

Sebuah perusahaan Asuransi memiliki banyak pegawai antara lain Sales, Satpam dan Manajer dengan spesifikasi sbb :

- a. Setiap pegawai memiliki NIP, nama, alamat, dan tahun masuk kerja
- b. Data satpam meliputi NIP, nama, alamat, gaji pokok dan jam lembur
- c. Bonus diberikan bagi Satpam yang bersedia untuk lembur dengan perhitungan Rp.10.000/jam lembur
- d. Data Sales meliputi NIP, nama, alamat, gaji pokok dan jumlah pelanggan yang berhasil direkrut
- e. Komisi diberikan dengan perhitungan Rp.50.000 untuk setiap pelanggan yang berhasil direkrut
- f. Data Manajer meliputi nama, alamat, gaji pokok, divisi dan tunjangan jabatan
- g. Tunjangan jabatan bagi Manajer diberikan jika telah bekerja selama 3 tahun sebesar 5% dari gaji pokok dan jika telah bekerja selama lebih dari 5 tahun sebesar 10% dari gaji pokok

Buatlah sebuah Kelas Pegawai yang menjadi induk Kelas Satpam, Kelas Sales dan Kelas Manajer. Lalu implementasikan konsep inheritance untuk menghitung gaji akhir pegawai melalui method **HitungGajiAkhir()** dan menampilkan informasi data pegawai perusahaan Asuransi tersebut pada tahun 2015.

#### Ketentuan:

- a. Data Satpam, Sales dan Manajer diinputkan melalui konstruktornya. Setelah itu semua data pegawai ditampilkan.
- b.  $Gaji\ Akhir = GajiPokok + Bonus\ atau\ Komisi\ atau\ Tunjangan$

#### Main Program dalam Bahasa Java

```
public static void main(String []args)
{
    Satpam S = new Satpam();
    Sales T = new Sales();
    Manajer M = new Manajer();
    S.setSatpam("Rendra", "0042", "Jl. Itik 15", 2000, 300000, 5);
    T.setSales("Wibisana", "0185", "Jl. Ayam 78", 2006, 500000, 10);
    M.setManajer("Adi", "0005", "Jl. Angsa 56", 1999, 1500000, "Keuangan");
    System.out.println("\n\n==DISPLAY DATA KARYAWAN==");
    S.cetakSatpam();
    T.cetakSales();
    M.cetakManajer();
}
```

Package Unguided

Pegawai.java

```

package Unguided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class pegawai {
    public String NIP;
    public String nama;
    public String alamat;
    public int tahunMasukKerja;

    protected String w,x,y;
    protected int z;

    public pegawai(String a, String b, String c, int d) {
        System.out.println("Konstruktor Point dijalankan ");
        w = a;
        x = b;
        y = c;
        z = d;
    }

    public void cetakpegawai() {
        System.out.println("Nama: " + nama);
        System.out.println("NIP: " + NIP);
        System.out.println("Alamat: " + alamat);
        System.out.println("Tahun Masuk: " + tahunMasukKerja);
    }
}

```

Satpam.java

```

package Unguided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class satpam extends pegawai {
    public int gajiPokok;
    public int jamLembur;
    public int totalGaji;
}

```



```

public satpam() {
    super(" ", " ", " ", 0);
    this.gajiPokok = 0;
    this.jamLembur = 0;
    System.out.println("konstruktor satpam dijalankan");
}

public void setSatpam(String nama, String NIP, String alamat, int
tahunMasukKerja, int gajiPokok, int jamLembur) {
    this.nama = nama;
    this.NIP = NIP;
    this.alamat = alamat;
    this.tahunMasukKerja = tahunMasukKerja;
    this.gajiPokok = gajiPokok;
    this.jamLembur = jamLembur;
}

public int hitungTotalGaji() {
    totalGaji = gajiPokok + (jamLembur * 10000);
    return totalGaji;
}

public void cetakSatpam(){
    super.cetakpegawai();
    System.out.println("Gaji Pokok : " + gajiPokok);
    System.out.println("Jam Lembur : "+ jamLembur);
    System.out.println("Gaji Akhir : "+ hitungTotalGaji());
}
}

```

Sales.java

```

package Unguided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class sales extends pegawai{
    public int gajiPokok;
    public int jumlahPelangganYangBerhasilDirekrut;
    public int totalGaji;
}

```

```

public sales(){
    super(" ", " ", " ", 0);
    gajiPokok = 0;
    jumlahPelangganYangBerhasilDirekrut = 0;
    System.out.println("Konstruktor sales dijalankan ");
}

    public void setSales(String nama, String NIP, String alamat, int
tahunMasukKerja, int gajiPokok, int
jumlahPelangganYangBerhasilDirekrut) {
    this.nama = nama;
    this.NIP = NIP;
    this.alamat = alamat;
    this.tahunMasukKerja = tahunMasukKerja;
    this.gajiPokok = gajiPokok;
    this.jumlahPelangganYangBerhasilDirekrut =
jumlahPelangganYangBerhasilDirekrut;
}

    public int hitungTotalGaji() {
        totalGaji = gajiPokok + ( jumlahPelangganYangBerhasilDirekrut *
50000);
        return totalGaji;
    }

    public void cetakSales(){
        super.cetakpegawai();
        System.out.println("Gaji Pokok : " + gajiPokok);
        System.out.println("Jam Lembur : "+
jumlahPelangganYangBerhasilDirekrut);
        System.out.println("Gaji Akhir : "+ hitungTotalGaji());
    }
}

```

#### Manager.java

```

package Unguided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class manager extends pegawai {
    public int gajiPokok;
}

```

```

public String divisi;
public int tunjangan;

public manager() {
    super(" ", " ", " ", 0);
    this.gajiPokok = 0;
    this.divisi = " ";
    System.out.println("konstruktor manager dijalankan");
}

public void setManager(String nama, String NIP, String alamat, int
tahunMasukKerja, int gajiPokok, String divisi) {
    this.nama = nama;
    this.NIP = NIP;
    this.alamat = alamat;
    this.tahunMasukKerja = tahunMasukKerja;
    this.gajiPokok = gajiPokok;
    this.divisi = divisi;
}

public int hitungGajiAkhir() {
    tunjangan = 0;
    int tahunKerja = 2015 - tahunMasukKerja;
    if (tahunKerja >= 5) {
        tunjangan = (int) (0.10 * gajiPokok); // 10% dari gaji pokok
    } else if (tahunKerja >= 3) {
        tunjangan = (int) (0.05 * gajiPokok); // 5% dari gaji pokok
    }
    return gajiPokok + tunjangan;
}

public void cetakManajer() {
    super.cetakpegawai();
    System.out.println("Gaji Pokok: " + gajiPokok);
    System.out.println("Divisi: " + divisi);
    System.out.println("Gaji Akhir: " + hitungGajiAkhir());
}
}

```

Main.java

```
package Unguided;
```

```

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class main {

    public static void main(String[] args) {
        satpam S = new satpam();
        sales T = new sales();
        manager M = new manager();

        S.setSatpam("Rendra", "0042", "Jl. Itik 15", 2000, 300000, 5);
        T.setSales("Wibisana", "0185", "Jl. Ayam 78", 2006, 500000, 10);
        M.setManager("Adi", "0005", "Jl. Angsa 56", 1999, 1500000,
"Keuangan");
        System.out.println("\n\n==DISPLAY DATA KARYAWAN==");

        S.cetakSatpam();
        System.out.println(" ");

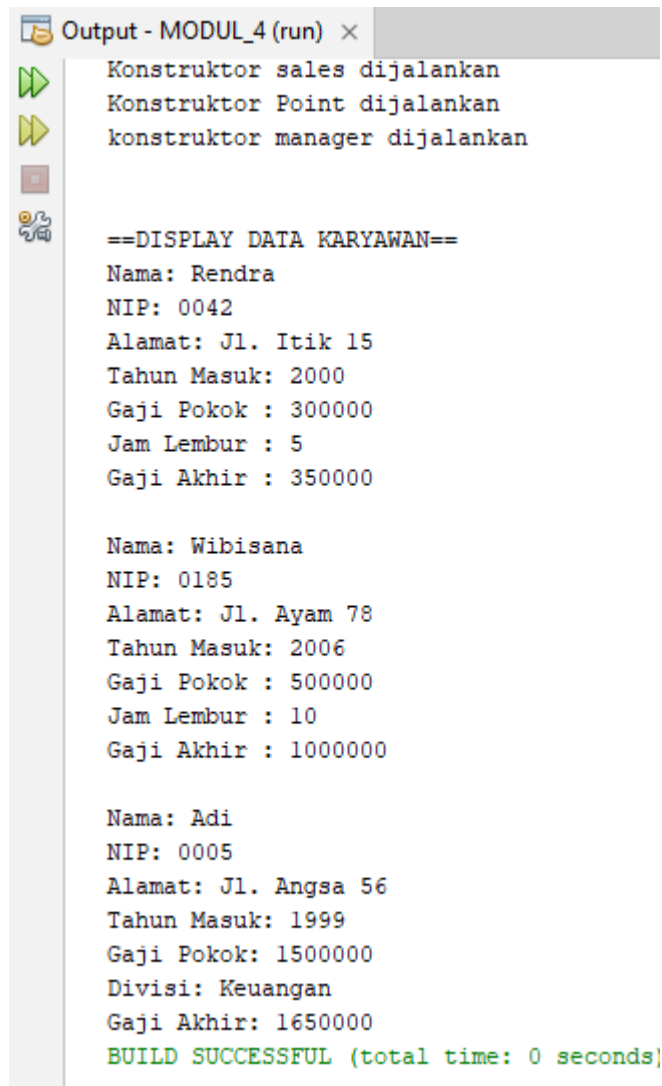
        T.cetakSales();
        System.out.println(" ");

        M.cetakManajer();
    }
}

```

Output :

```
Output - MODUL_4 (run) ×  
run:  
Konstruktor Point dijalankan  
konstruktor satpam dijalankan  
Konstruktor Point dijalankan  
Konstruktor sales dijalankan  
Konstruktor Point dijalankan  
konstruktor manager dijalankan  
  
==DISPLAY DATA KARYAWAN==  
Nama: Rendra  
NIP: 0042  
Alamat: Jl. Itik 15  
Tahun Masuk: 2000  
Gaji Pokok : 300000  
Jam Lembur : 5  
Gaji Akhir : 350000  
  
Nama: Wibisana  
NIP: 0185  
Alamat: Jl. Ayam 78  
Tahun Masuk: 2006  
Gaji Pokok : 500000  
Jam Lembur : 10  
Gaji Akhir : 1000000  
  
Nama: Adi  
NIP: 0005  
Alamat: Jl. Angsa 56  
Tahun Masuk: 1999  
Gaji Pokok: 1500000
```



```
Output - MODUL_4 (run) x
Konstruktor sales dijalankan
Konstruktor Point dijalankan
konstruktor manager dijalankan

==DISPLAY DATA KARYAWAN==
Nama: Rendra
NIP: 0042
Alamat: Jl. Itik 15
Tahun Masuk: 2000
Gaji Pokok : 300000
Jam Lembur : 5
Gaji Akhir : 350000

Nama: Wibisana
NIP: 0185
Alamat: Jl. Ayam 78
Tahun Masuk: 2006
Gaji Pokok : 500000
Jam Lembur : 10
Gaji Akhir : 1000000

Nama: Adi
NIP: 0005
Alamat: Jl. Angsa 56
Tahun Masuk: 1999
Gaji Pokok: 1500000
Divisi: Keuangan
Gaji Akhir: 1650000
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Program diatas adalah program pegawai pekerja di suatu instansi asuransi dengan 5 file java yaitu pegawai,satpam,sales,manager, dan program utama agar lebih mudah dan terorganisir. File pegawai berfungsi sebagai template untuk parent dari ke 3 file lainnya yaitu satpam, sales, dan manager. Pemanggilan class parent dengan konstruktor dan terdapat ketentuan pada program utama harus memasukan lewat mutator aga sedikit tricky dengan mengosongkan parameter dan sintaks super untuk memanggil class parent dikosongkan semua agar bisa dimasukan dengan fungsi mutator. Dibawah ini adalah contoh dari salah satu class

```

public satpam() {
    super(" ", " ", " ", 0);
    this.gajiPokok = 0;
    this.jamLembur = 0;
    System.out.println("konstruktor satpam dijalankan");
}

public void setSatpam(String nama, String NIP, String alamat, int ta
    this.nama = nama;
    this.NIP = NIP;
    this.alamat = alamat;
    this.tahunMasukKerja = tahunMasukKerja;
    this.gajiPokok = gajiPokok;
    this.jamLembur = jamLembur;
}

```

Pada persegi merah terlihat konstruktor yang tidak ada parameternya dan sintaks supernya yang di kosongkan. Demi agar method setsatpam bisa mengisi nilai pada member class satpam agar sesuai dengan ketentuan soal yang seperti dibawah ini

```

public static void main(String []args)
{
    Satpam S = new Satpam();
    Sales T = new Sales();
    Manajer M = new Manajer();
    S.setSatpam("Rendra","0042","Jl. Itik 15",2000,300000,5);
    T.setSales("Wibisana","0185","Jl. Ayam 78",2006,500000,10);
    M.setManajer("Adi","0005","Jl. Angsa 56",1999,1500000,"Keuangan");
    System.out.println("\n\n==DISPLAY DATA KARYAWAN==");
    S.cetakSatpam();
    T.cetakSales();
    M.cetakManajer();
}

```

